論文 86-23-3-7

RNS를 이용한 벡터 座標 回轉 演算 프로세서

(A Vector-Coordinate-Rotation Arithmetic Processor Using RNS)

趙 源 敬*, 林 寅 七**

(Won Kyung Cho and In Chil Lim)

要 約

本 論文에서는 RNS(Residue Number Systems)를 이용하여 벡터의 座標 回轉 演算 프로세서와 sine과 cosine의 근사값을 구하는 方法을 提案하였다. 提案된 알고리즘은 CORDIC 알고리즘 보다 高速의 回轉 演算을 실행할 수 있다.

시뮬레이션 結果, sine과 cosine의 값의 平均 誤差는 0.0025였고 回轉 演算의 平均 誤差는 0.65였다.

또한 提案된 프로세서는 同一한 構造의 演算表의 排列에 의하여 構成되기 때문에 設計가 용이하고 集積回路로 製作하기가 容易하다.

Abstract

This paper shows that we can design a vector-coordinate rotation processor and obtain the approximate evaluations of sine and cosine based upon the use of residue number systems. The algorithm results in the considerable improvement of the computation speed when compared to CORDIC algorithm.

The results from computer simulation show that the mean error of sine and cosine is 0.0025 and the mean error of coordinate rotation arithmatic is 0.65. Also, the proposed processor has the efficiency for the design and fabrication of integrated circuit, because it consists of the array of idecntially structured look-up tables.

Ⅰ. 序 論

最近, 並列處理가 가능하고 回路 設計 方法이 組織的인 RNS(Residue Number Systems)을 이용한 演算回路의 設計에 관한 硏究가 활발하다.[1]

RNS를 이용한 디지탈 回路의 設計 方法은 1967년 Szabo와 Tanaka에 의하여 基本 理論이 확립되었다.[2] 현재에는 Taylor, Jenkins 等에 의하여 디지탈 信號處理用 乘算器,[3] 디지탈 필터,[4] FFT,[5] 等의 回路 設計에 RNS가 주로 이용되고 있다.

直角 座標係에서 벡터 R의 성분 X, Y와 回轉角 θ가 주어진 경우 座標 回轉 演算을 위해서는 $\cos\theta$ 와 $\sin\theta$의 값을 구하여야 하고 4回의 乘算과 2回의 加·減算이 필요하다.

既存의 디지탈 信號 處理 알고리즘에서는 一般的으로 乘算의 回路 構成이 복잡하고 動作 速度가 느리기 때문에 乘算을 줄이거나 생략할 수 있는 方法을 摸索하고 있다.[6]

CORDIC(Coordinate Rotation Digital Computer)[7]에서는 乘算을 shift와 加算으로 변환하여 간단한 回路 構成으로 回轉 演算을 실행할 수 있지만 演算 알

*正會員 慶熙大學校 工科大學 電子工學科
(Dept. of Elec. Eng., Kyunghee Univ.)
**正會員 漢陽大學校 工科大學 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1986年 3月 3日

고리즘이 逐次形이므로 高速 演算은 불가능하다.

그러나 RNS에서는 乘算과 加·減算의 演算 速度가 같고 이들의 演算이 同一한 크기의 演算表(look-up table)에 의하여 실행되기 때문에 高速의 演算回路를 構成할 수 있다.[2]

RNS에서는 整數만을 취급하기 때문에 $\cos\theta$, $\sin\theta$의 演算은 $\cos\theta$와 $\sin\theta$에 常數 K를 곱하여 整數로 變換하여 벡터의 回轉 演算을 행한다. 그러므로 回轉演算에 필요한 乘算에서 두 變數의 곱의 結果는 1/K로 스케일링 되어야 하므로 1/K의 스케일링을 포함하는 乘算器를 이용해야 한다.

本 論文에서는 RNS를 이용하여 sine, cosine 값을 구하는 回路와 QSA(Quarter Square Algorithm)을 이용한 스케일링 乘算器를 이용하여 벡터 座標 回轉 演算 回路를 設計하였다.

提案된 回路는 파이프라인(pipe-line) 方式으로 構成되어 4개의 클럭 펄스마다 1回씩 回轉 演算을 高速으로 실행할 수 있다.

Ⅱ. 座標回轉 演算을 위한 스케일링 乘算

1. 座標回轉 演算의 基本式

벡터 R이 直角 座標係에서 X, Y로 표시될 경우, R을 θ만큼 回轉한 R'의 좌표 X', Y'는

$$X' = X\cdot\cos\theta \mp Y\cdot\sin\theta$$
$$Y' = Y\cdot\cos\theta \pm X\cdot\sin\theta \qquad (1)$$

이다.

(1)式을 계산하기 위해서는 $\sin\theta$, $\cos\theta$ 값과 4回의 乘算과 2回의 加·減算을 필요로 한다. 그러나 RNS에서는 演算表를 이용하여 加·減算과 同一한 速度로 乘算을 실행할 수 있기 때문에, 本 論文에서는 RNS를 사용한 乘算器를 이용한다.

또한, $\sin\theta$와 $\cos\theta$의 근사값을 구하기 위한 回路를 RNS를 이용하여 構成한다.

2. RNS를 이용한 스케일링 乘算器

1) RNS의 基本式

서로 素의 集合을 $m=(m_1, m_2, \cdots, m_L)$이라하면 M은 (2)式과 같이 나타낸다.

$$M=\prod_{i=1}^{L} m_i \qquad (2)$$

어떤 수 X가 [0, M) 범위의 整數이면 X는 (3)식과 같이 나타낼 수 있다.

$$X=k_i\cdot m_i + x_i \qquad (3)$$

(3)식에서 x_i를 x의 i차 유수(residue)라 하고 $|X|_{m_i}$ 또는 $X_{mod\,m_i}$로 표시한다. 그러면 $X\cong(x_1, x_2, \cdots, x_L)$와 같이 X를 residue로 표시할 수 있다.

RNS에서 加·減·乘算을 ○로 표시하면 整數 X, Y의 演算은

$$(z_1, z_2, \cdots, z_L) = (x_1, x_2, \cdots, x_L) \circ (y_1, y_2, \cdots, y_L)$$
$$z_i = (x_i \circ y_i) \bmod m_i \qquad (4)$$

가 성립한다. 윗 식에서 z_i는 x_i와 y_i만에 의한 연산이므로 演算表의 용량이 작게되어 演算表를 이용한 디지탈 演算 回路의 設計에 RNS가 적합하다.

RNS에서 음수의 표현이 필요한 경우는 [0, M/2)의 범위를 양의 정수로, [M/2, M)을 음의 정수로 표현하여 처리한다.[8]

2) 스케일링을 포함한 乘算器

두 整數의 곱을 1/K로 스케일링한 결과를 얻는 승산기를 설계하기 위하여 Taylor의 알고리즘[1]을 변형하여 사용한다.

두 정수의 곱을 QSA(Quarter Square Algorithm)으로 표시하면

$$x\cdot y = \phi(x+y) - \phi(x-y)\,;\ \phi(s) = (s/2)^2 \qquad (5)$$

x와 y가 n비트로 표시되는 정수이고 x·y의 연산 결과를 직접 연산표에 기록할 경우, 2^{2n}W(words)가 필요하나 (5)식을 이용하면

$$0 \le x+y < 2^{n+1}$$
$$-2^n < x-y < 2^n \qquad (6)$$

으로 되어 $2\cdot 2^{n+1}$W의 연산표가 필요하게 된다.

RNS를 이용하여 (5)식을 계산하는 과정은 다음과 같다. 두 변수 x, y의 범위가 [0, M/2)인 경우 스케일링 상수 K를 K=INT(M/4)로 취하면

$$z = x\cdot y/K = \phi(x+y)/K - \phi(x-y)/K \qquad (7)$$

이 된다. 윗 식을 RNS로 표시하면

$$\phi_i(x+y)/K = |\,(x+y)^2/M\,|_{m_i}$$
$$\phi_i(x-y)/K = |\,(x-y)^2/M\,|_{m_i} \qquad (8)$$

여기서 i=1, 2, ···, L

이 되어 (8)식의 결과를 연산표에 기록시켜 그림(1) 과 같이 1/K배의 스케일링 연산을 포함한 乘算器를 구성할 수 있다.

그림(1)의 乘算 回路에서 변수 X는 入力 整數이다. $\sin\theta$ 또는 $\cos\theta$를 K배한 整數값을 Y이라 하면

$$Y = INT(K\cdot\cos\theta) \text{ 또는}$$
$$Y = INT(K\cdot\sin\theta) \qquad (9)$$

와 같이 Y은 실수에 常數 K를 곱하여 정수 부분을 취한 것이 되어 (8)식은 整數의 演算으로 된다.

그 결과, 그림(1)의 出力, $|Z|_{m_i}$는 $X\cdot\cos\theta$ 또는 $X\cdot\sin\theta$의 연산이 되어 (1)식에서 필요한 基本乘算이 RNS를 이용하여 가능하게 된다.

이때, $0\le\sin\theta,\ \cos\theta\le 1$이므로 Z는 $0\le Z\le X_{max}$로 되어 승산에서 orerflow가 발생되지 않는다.

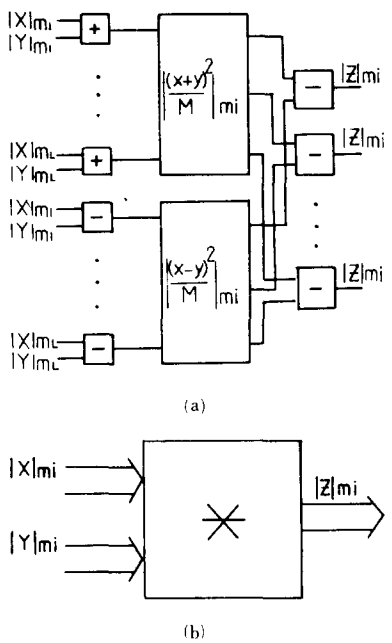

그림 1. RNS를 이용한 스케일링 승산기
(a) QSA를 이용한 스케일링 승산기
(b) 블록 다이아그램
Fig. 1. Scaling multiplier using RNS.
(a) Scaling multiplier usinig QSA.
(b) Block diagram.

3. RNS를 이용한 三角函數 發生.

RNS를 이용하여 $\cos\theta$와 $\sin\theta$의 近似값을 發生시키기 위하여 그림(2)와 같이 $\sin\theta$와 $\cos\theta$를 n개의 區間으로 나누어 1次 函數로 近似값을 구한다.

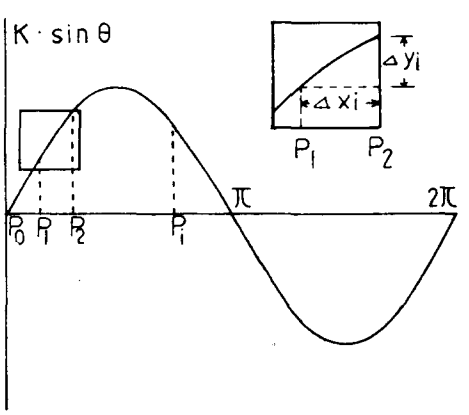

그림 2. 1차 함수에 의한 sine 함수의 근사화.
Fig. 2. Approximation of sine funtion by first-order function.

$\theta(0\leq\theta\leq 2\pi)$를 2^{B-3}배한 값의 整數 부분을 θ_B라 하면

$$\theta_B = a_{B-1}\cdot 2^{B-1} + a_{B-2}2^{B-2} + \cdots\cdots + a_0\cdot 2^0 \qquad (10)$$

(10)식에서 上位 B_1비트의 값을 θ_L이라 하고 下位 B_2 $(=B+3-B_1)$ 비트의 값을 θ_R이라 하면

$$\theta_B = \theta_L + \theta_R \qquad (11)$$

윗 식에서 2π를 θ_L 等分한 각 점의 값을 P_i라 하고 각 점에서의 기울기를 C_i라 하면

$$K\cdot\sin\theta = P_i + C_i\cdot\theta_R, \quad i=0, 1, \cdots, \theta_L \qquad (12)$$

식과 같이 된다. 윗 식에서 C_i는

$$C_i = \frac{y_{i+1}-y_i}{x_{i+1}-x_i} = \frac{\triangle y_i}{\triangle x_i} \qquad (13)$$

이다. (13)식을 (12)식에 대입하면

$$K\cdot\sin\theta = P_i + \triangle y_i\cdot\theta_R\cdot\frac{1}{\triangle x_i} \qquad (14)$$

로 되어 $\triangle x_i$에 의한 스케일링 演算이 필요하다. 이때 $\triangle x_i$는 $\triangle x_i = 2^{B_2}$로 되어 B_2에 의하여 결정되는 常數이기 때문에 RNS에서 스케일링 演算이 가능하게 된다.

RNS에서의 스케일링은 다음과 같다. X를 入力이라 하고 Y를 스케일링 결과, S를 스케일링 인수라 하면

$$Y = \left\lfloor \frac{X}{S} \right\rfloor \qquad (15)$$

여기서 $\lfloor\ \rfloor$는 ○의 정수값을 의미한다. 윗 식을 다시 쓰면 $X = Y\cdot S + |X|_S$이므로

$$Y = \frac{X - |X|_S}{S} \qquad (16)$$

이다. S의 곱의 역인수(multiplicative inverse), $\left|\frac{1}{S}\right|$을 다음과 같이 정의하면

$$|S\cdot|\frac{1}{S}|_{m_i}|_{m_i} = 1 \qquad (17)$$

Y의 i는 유수(residue) y_i는

$$y_i = |Y|_{m_i} = ||X - |X|_S|_{m_i}\cdot|\frac{1}{S}|_{m_i}|_{m_i} \qquad (18)$$

과 같이 구할 수 있다.

$\cos\theta$값도 $\sin\theta$와 같은 方法으로 구할 수 있다. 그림(3)은 RNS를 이용하여$\sin\theta$, $\cos\theta$값을 구하는 回路의 構成圖이다. 그림(3)에서 P_i와 $\triangle y_i$값을 연산표를 이용하여 구하고 다른 부분은 RNS의 연산을 이용하였다.

Ⅲ. 座標回轉 演算을 위한 回路의 構成

RNS를 이용하여 벡터의 座標回轉 演算을 실행하는 회로의 全体 構成은 그림(4)와 같다.

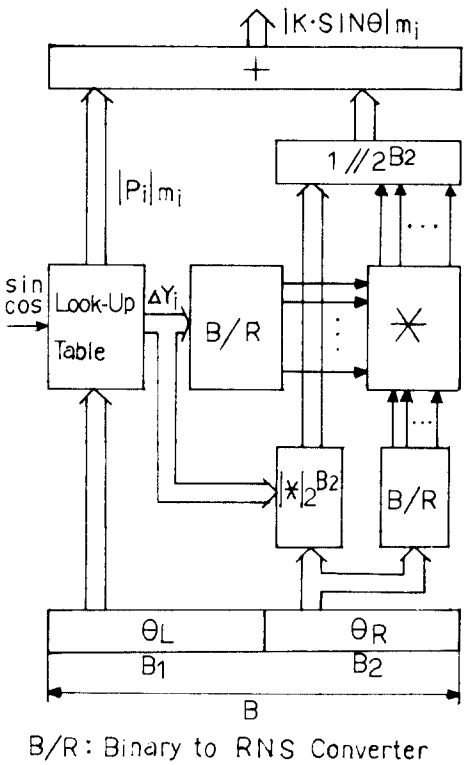

그림 3. SIN(θ)와COS(θ)의 회로.
Fig. 3. Generator of SIN(θ) and COS(θ).

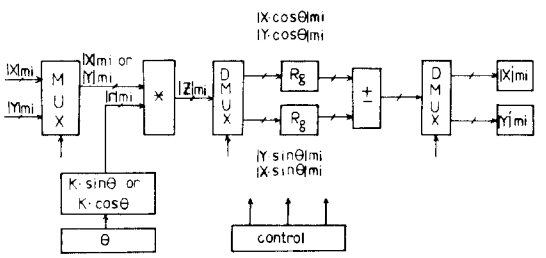

그림 4. 벡터 좌표 회전 회로의 구성도.
Fig. 4. Configuration of vector coordinate rotation circuits.

그림(4)에서 각 부분의 演算 回路는 RNS를 이용한 연산표를 이용하고 있기 때문에 전체 回路 構成이 ROM과 Rg(Register), Mux(mutiplexer), DMUX (de-multiplexer)만으로 구성되어 組織的인 設計가 가능하다.

또한, 各 乘算의 結果가 每 클럭마다 Rg에 보관되고 信號의 흐름이 한 方向으로만 진행되므로 파이프라인(pipe-line) 處理가 가능하고 4 개의 클럭 펄스마다 1回의 回轉 演算이 가능하다.

그림(4)의 回路 構成에 필요한 ROM의 용량은 入力 整數 X, Y의 크기에 따른 모듈리(moduli) 選擇과 正弦과 餘弦函數를 발생시키기 위한 연산표의 용량에 관계된다.

Ⅳ. 誤差 및 시뮬레이션 結果

m=(11, 13, 15)인 경우를 例로 들어 디지탈 컴퓨터로 시뮬레이션 하였으며 θ_L과 θ_R의 選擇에 따른 誤差를 考察한다.

m을 위와 같이 선택하면 M=2145이고 K=536이 되어 X와 Y의 最大값은 536이 된다. 이때 $K\cdot\sin\theta$와 $K\cdot\cos\theta$의 最大값도 K와 같다.

θ를 B비트로 표시하고 θ_L과 θ_R을 각각 B_1, B_2비트로 표시한다.

이때 B와 B_1의 選擇에 따라 그림(3)의 回路에서 發生된 오차는 그림(5)와 같다. 그림(5)의 결과에서 平均 誤差 0.0025 이내의 三角函數를 發生시키기 위해

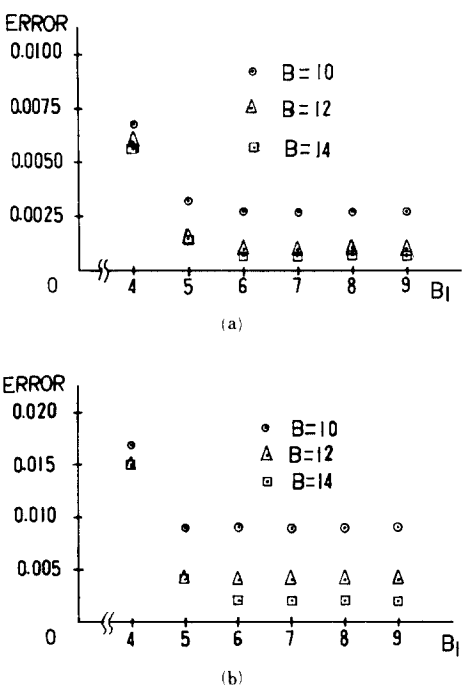

그림 5. SIN(θ), COS(θ)의 오차
(a) B와 B_1에 따른 평균 오차
(b) 계산 결과의 98%를 포함하는 오차의 범위

Fig. 5. Errors of SIN(θ) and COS(θ).
(a) Means of errors with B and B_1.
(b) Range of errors including 98% of computing results with B and B_1.

서는 B=12, B_1=7인 경우가 적합함을 알 수 있다. 또한, B=12이고 B_1=7일때 三角函數 發生 결과의 98%를 포함하는 誤差 범위는 0.004임을 알 수 있다.

B=12, B_1=7인 경우에 발생된 正弦과 餘弦 函數를 이용하여 回轉 演算을 한 結果의 오차는 그림(6)과 같다. 그림(6)에서 平均 誤差는 0.65이고 계산 결과의 98%를 포함하는 오차의 범위는 1.8로서 提案된 벡터 座標 回轉 回路는 실제의 그래픽 데이타의 回轉 演算에 이용할 수 있음을 확인하였다.

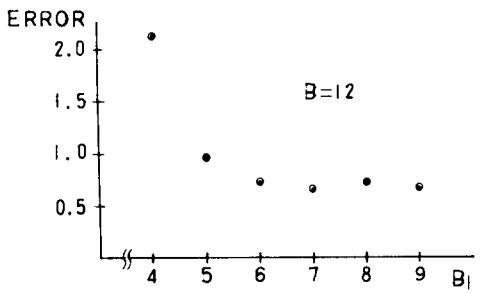

그림 6. B=12일때 B_1에 따른 회전 결과의 98%를 포함하는 오차 범위

Fig. 6. Range of errors including 98% of computing results with B_1(B=12).

V. 結 論

RNS를 이용하여 벡터의 座標回轉을 高速으로 실행할 수 있는 演算 回路의 設計 方法을 提案하였다.

提案된 回路는 파이프라인 處理에 의하여 4 개의 클럭 펄스마다 1回씩 벡터의 座標回轉 演算을 할 수 있다.

CORDIC 알고리즘[6]을 이용하여 座標回轉 演算을 할 경우, 12bits의 해상도를 위해서는 24클럭 펄스가 필요하나 提案된 알고리즘에서는 4 클럭 펄스만이 필요하게 되어 고속의 벡터 座標回轉 演算이 가능하다.

그리고 回路의 大部分이 演算表의 排列로서 이루어지므로 設計가 容易하고 메모리 製造 技術의 發達로 그림(4)의 回路를 VLSI로 製作하기가 容易하다.

컴퓨터 시뮬레이션 結果, sine과 cosine의 값은 平均 誤差 0.0025이내로 구할 수 있었고, 回轉 演算은 m=(11, 13, 15)로 選擇했을 경우 平均 誤差가 0.65였다.

그러므로 위와 같은 정도의 誤差를 許容하는 범용의 디지탈 信號 處理와 그래픽 디스플레이 데이타(graphic display data)의 處理를 위한 演算 回路에 提案된 回路를 이용할 수 있다.

Ⅱ. 절에서 提案된 스케일링 演算을 포함한 乘算器를 그래픽 데이타의 스케일링이 가능하도록 擴張하여 그래픽 處理에 필요한 位置 變換(translation), 스케일링(scaling), 回轉(rotation) 演算을 함께 處理할 수 있는 그래픽 프로세서를 設計하는 硏究가 필요하다.

參 考 文 獻

[1] F.J. Taylor, "An Overflow-Free Residue Multiplier," *IEEE Trans. Comput.*, vol. C-32, no. 5, pp. 501-507, May. 1983.

[2] N.S. Szabo and R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, New York: McGraw-Hill, 1967.

[3] F.J. Talyor, "A VLSI Residue Multiplier," *IEEE Trans. Comput.*, vol. C-31, no. 6, pp. 540-546, Jun. 1982.

[4] W.K. Jenkins and B.J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filter," *IEEE Trans. Circuits and Syst.*, vol. CAS-24, no. 4, pp. 191-201, Apr. 1977.

[5] Ben-Dau Tseng, et al, "Implementation of FFT Structures Using the Residue Number System," *IEEE Trans. Comput.*, vol. C-28, no. 11, pp. 831-844, Nov. 1979.

[6] A.M. Despain, "Fouier Transform Computers Using CORDIC Iterations," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 993-1001, Oct. 1974.

[7] G.L. Haviland and A.A. Yuszynski, "A CORDIC Arithmetic Processor Chip," *IEEE Trans. Comput.*, vol. C-29, no. 2, pp. 68-79, Feb. 1980.

[8] Thu Van Vu, "Efficient Implementation of Chinese Remainder Theorem for Sign Detecition and Residue Decoding," *IEEE Trans. Comput.*, vol. C-34, no. 7, pp. 646-651, Jul. 1985.

[9] F.J. Taylor and C.H. Huang, "An autoscale Residue Multiplier," *IEEE Trans. Comput.*, vol. C-31, no. 4, pp. 321-325, Apr. 1982.

[10] G.A. Jullien, "Residue Number Scaling and Other Operations Using ROM Arrays," *IEEE Trans. Comput.*, vol. C-27, no. 4, pp. 325-336, Apr. 1978.

[11] J.D. Foley and A.V. Dam, *Fundamentals of Ierative Computer Graphics*, Addison-Wesley, 1982. *